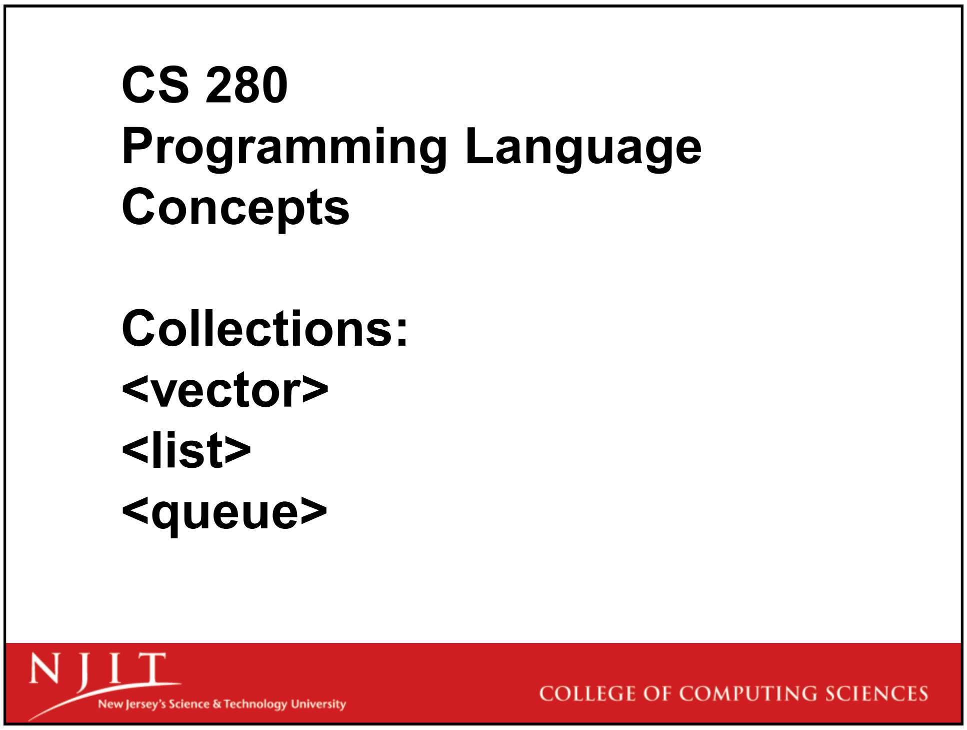
NJIT

New Jersey's Science & Technology University

THE EDGE IN KNOWLEDGE



CS 280 Programming Language Concepts

Collections: <vector> <list> <queue>

NJIT New Jersey's Science & Technology University

COLLEGE OF COMPUTING SCIENCES

Standard Template Library

- C++ provides templated types for collections
- The templates come with methods to perform operations on the collection
- The type of what is in the collection is part of the declaration. That means that it is a compile-time parameter
- The collections are thus strongly typed and type-safe for what they contain

Vectors

- Available via #include <vector>
- A vector is an array of adjustable size
- Constant time access to entries in the vector
- Adding an item to the end of a vector might cost time to reallocate memory and copy information
- Removing an item from the end of a vector is constant time
- Adding item(s) to the middle of a vector might cost time to reallocate memory, definitely costs time to adjust the contents of the vector to "make room" for the new item(s)
- Removing item(s) from middle of the vector will cost time to adjust the contents of the vector to "close the gap" created by removed items

Vectors

- The type of the members of the array is given inside of < > at declaration:
  - `vector<int>` is a vector of `int`
- Comes with methods to modify members of the vector, determine size, etc.
- The operator [ ] is overloaded to provide access to members of the vector, but [ ] can only access members that are already in the vector.
- Use the at() method for vector access that does range checking
- push_back() method adds to the end of the vector

NJIT New Jersey's Science & Technology University COLLEGE OF COMPUTING SCIENCES

Vector example

```
vector<int> Myvec;

// this line would fail
//  because there is no [0] yet
Myvec[0] = 10;

// but these lines are ok
Myvec.push_back(10); // add it
    // Myvec[0] is ok now!
Myvec[0] = 100; // change it
```

NJIT New Jersey's Science & Technology University COLLEGE OF COMPUTING SCIENCES

Looking at the whole vector

```
// treat it like an array

vector<int> Myvec;
for(int i=0; i<Myvec.size(); i++)
     cout << i << ":" << Myvec[i] << endl;
```

COLLEGE OF COMPUTING SCIENCES

Looking at the whole vector

```
// use an iterator

vector<int> Myvec;
vector<int>::iterator it;
for(it = Myvec.begin(); it != Myvec.end(); it++ )
         cout << ":" << *it << endl;
```

COLLEGE OF COMPUTING SCIENCES

Looking at the whole vector

```
// use range-based for loops (C++ 11)
vector<int> Myvec;
for(int i : Myvec )
         cout << ":" << i << endl;

// or…
for(int& ir : Myvec )
         cout << ":" << ir << endl;
```

Finding something in a vector

- Vectors have a find() method
- The type of the argument to find is the type of the content of the vector (so, for example, if I have a vector of int, find() takes an int)
- The complexity of find is the complexity of find in an array
- Find returns an iterator
- If the find fails, the value of the iterator is == end() of the vector

list

- Available via #include <list>
- As with a vector, a list must indicate the type of the elements in the list when it is declared. Therefore list<int> is a list of integers
- operator[] is NOT overloaded on lists
- Unlike a vector, inserting and deleting elements from the list is a constant time operation
- Iterators are available to sequence through the list
- find() methods also return an iterator
  - if find() fails, it returns the end() of the collection

Queues

- Available via #include <queue>
- The container implements a FIFO (first in, first out) queue
- Add element with push() method
- Remove element with pop() method
- There are no iterators

NJIT

THE EDGE IN KNOWLEDGE